INS FORAT DEL VENT– INTERNATIONAL BACCALUREATE

INFORMÀTICA I - POO

TEMA 6: INTRODUCCIÓ A LA POO

TEMA 6 PROGRAMACIÓ ORIENTADA A OBJECTES

BASE TEÒRICA

Ara entrarem a fons en allò que constitueix un nou paragigma de la programació : les **classes**, i per tant, la **programació orientada a objectes (POO)**. Així que prepara't per canviar la mentalitat, i l'enfocament de la programació tal com ho hem vist fins ara.

Introduirem nous conceptes que normalment s'associen a la programació orientada a objectes, com són: objecte, missatge, mètode, classe, herència, interfície, etc.

Cal definir alguns conceptes clau en la programación orientada a objetos:

- **POO**: Sigles de *"Programació Orientada a Objectes"*. En anglès s'escriu al revés OOP. La idea bàsica d`aquest paradigma de programació és agrupar les dades i les funcions per englobar-los en una única entitat: l'objecte. Cada programa és un objecte, que està format per objectes que es relacionen entre ells.
  Això no vol dir que la idea de la programació estructurada hagi desaparegut, de fet es reforça i resulta més evident, com comprovaràs quan vegem conceptes com l'herència.

- **Classe**: Una classe es pot considerar un patró, una plantilla per construir objectes. Es diu tècnicament que una clase és l'especificació de un tipus de dada, és simplement una declaració i el seu codi associat.

- **Objecte**: Un objecte és una unitat que engloba dades i procediments necessaris per al tractament d'aquestes dades, és una instància (un exemplar) d'una classe determinada.
  Fins ara havíem fet programes on les dades i les funcions estaven perfectament separades, quan es programa amb objectes això no és així, cada objecte conté dades i funcions. I un programa es construeix com un conjunt d'objectes, o fins i tot com un únic objecte.

- **Missatge**: El missatge és la manera com es comuniquen i interrelacionen els objectes entre si. Un missatge no és més que una crida a una funció d'un objecte determinat. Quan cridem una funció d'un objecte, sovint direm que estem enviant un missatge a aquest objecte. En aquest sentit, missatge és el terme adequat quan parlem de programació orientada a objectes en general.

- **Mètode**: Es tracta d'un altre concepte de POO, els missatges que arribin a un objecte es processaran executant un mètode determinat d'aquest objecte. Un mètode és una funció o procediment que pertany a un objecte.

- **Herència**: És la capacitat de crear noves classes basant-se en classes prèviament definides, de les quals s'aprofiten certes dades i mètodes, se'n rebutgen d'altres i se n'afegeixen de noves.
  Serà possible dissenyar noves classes basant-se en classes ja existents, això s'anomena derivació de classes, en POO, herència. Quan se'n deriva una classe d'altres, normalment s'hi afegiran nous mètodes i dades.

- **Interface**: Es tracta de la part de l'objecte que és visible per a la resta dels objectes. És a dir, que és el conjunt de mètodes (i de vegades dades) de què disposa un objecte per comunicar-s'hi. Les classes i per tant també els objectes definits a partir d'elles tenen parts públiques i parts privades. Generalment, anomenarem la part pública d'una classe o objecte la interfície.

- **Declaració d'una classe**

La primera paraula que apareix és lògicament ***class*** que serveix per definir una classe i per declarar objectes d'aquesta classe.

```
[public] class <identificador de classe> {
        <llista de atributs i mètodes>
};
```

La llista d'atributs i mètodes será:

- Les dades (*atributs*) es declaren de la mateixa manera que ho fèiem fins ara, llevat que no poden ser inicialitzades, recorda que parlem de declaracions de classes i no de definicions d'objectes.

- Els *mètodes* (funcions) poden ser simplement declaracions de prototips, que cal definir a part de la classe i poden ser també definicions.

- El modificador public és opcional i indica que la classe és accesible des de qualsevol programa. Si no es posa només ho será des de les classes dins del paquet (package), que és un conjunt de classes agrupades.

Per tant, una classe és una agrupació de dades (variables o atributs) i de funcions (mètodes) que operen sobre aquestes dades. Un exemple d'una classe:

```
public class Punt {
        public double x;
        public double y;
        public Punt(double a, double b) {
                x = a;
                y = b;
        }
        public double calcularDistancia() {
                double z;
                z=Math.sqrt((x*x)+(y*y));
                return z;
        }
}
```

- **Elements d'un classe**

  - **Atributs**

Els atributs defineixen l'estructura dels objectes que instancien una classe. Cada objecte que es crea d'una classe té la seva pròpia copia dels atributs i dels mètodes (recordem que una classe és una plantilla). Per exemple, cada objecte de la classe Punt te les seves pròpies coordenades x i y. Per accedir-hi al valor d'un atribut escriurem:

Objecte.nom_atribut*;*

Per exemple :
```
p1.x=5;
p1.y=7;
System.out.println(p1.x+" , "+p1.y);
```

Per aplicar un mètode a un objecte concret farem: ***Objecte.nom_mètode()****;*

Per exemple:
```
double d = p1.calcularDistancia();
System.out.println("distancia: " + d);
```

- **Mètodes**

Constructors

El primer mètode que s'invoca per a crear un objecte és el constructor. La principal missió del constructor es reservar memòria i inicialitzar els atributs d'un objecte creat a partir d'una classe. No tenen valor de retorn (ni void) i no s'hereden, i s`han d'anomenar igual que la classe. Una classe pot tenir varis constructors, que es diferenciaran pel tipus i nombre d'arguments (fet que s'anomena *sobrecarrega*). Es diu constructor per defecte al constructor que no te arguments, i quan no hi ha cap constructor definit, el compilador crea un per defecte.

Modificadors (Setters)

Un objecte aïlla certs atributs de l'exterior y nomes es poden accedir o modificar mitjançant mètodes públics de lectura (get) o de escriptura (set). Cada objecte és una entitat encapsulada. En la classe es defineix la forma d'encapsulament que tindran els objectes mitjançant el nivell d'ocultament dels seus membres. Aquests nivells s'estableixen amb els especificadors d'accés, que són:

- De lliure accés (**public**). Els mètodes de qualsevol classe tenen accés. Es denota amb "+".
- D'accés predeterminat (per defecte o sense especificador, **package**), tenen accés els mètodes de les classes del mateix paquet.
- D'accés protegit, només poden accedir els mètodes de la pròpia classe i les seves classes derivades (**protected**), Es denota amb "#".
- De accés **private**, tenen accés els mètodes de la pròpia classe. Es denota amb "-".

Els mètodes modificadors són els que permetran accedir a un atribut de la classe i modificar el seu valor.

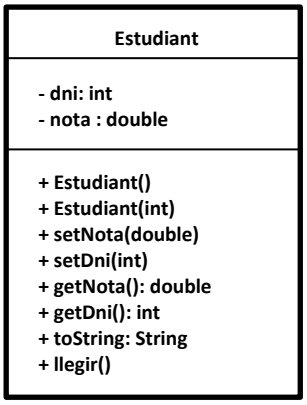
Accesors (getters)

Seran els mètodes mitjançant els quals es podrà accedir al valor d'un atribut de la classe. Sempre actuaran sobre la pròpia classe (paràmetre implícit).

Entrada i/o sortida

Seran els mètodes que permetran fer operacions d'entrada i sortida de dades, com per exemple introduir valors als atributs des de el teclat o mostrat valors dels atributs per pantalla.

Podem veure un exemple, la classe Estudiant, a partir de la seva representació UML:



La implementació de la classe Estudiant serà:

```
public class Estudiant {
   //Atributs de la classe
   private int dni;
   private double nota;

   //Mètodes
   //Constructor per defecte
   public Estudiant() {
      this.dni = 0;
      this.nota = -1;
   }
   //Constructor
   public Estudiant(int dni) {
      this.dni = dni;
      this.nota = -1;
   }
   //Consultors
   public void setNota(double nota) {
      this.nota = nota;
   }
   public void setDni(int dni) {
      this.dni = dni;
   }
   //Modificasdors
   public double getNota() {
      return nota;
   }
   public int getDni() {
      return dni;
   }
```

```
    //Entrada i sortida
    @Override
    public String toString() {
        if(nota!=-1) return ("DNI: " + dni + "  Nota: " + nota);
        else return ("DNI: " + dni + "  Nota: NP");
    }
    public void llegir() {
        Scanner teclat = new Scanner (System.in);
        System.out.println("Escriu el dni de l'alumne: ");
        int dni = teclat.nextInt();
        System.out.println("Escriu la nota de l'alumne (-1 si es NP): ");
        double aux = teclat.nextDouble();
        if(aux>=0 && aux <= 10) nota = aux;
        else nota = -1;
    }
```

Un objecte és una instància d'una classe, es una cosa tangible o visible, que pot ser entès, sobre el qual es pot aplicar alguna acció. També es considerat com un exemplar concret de la classe. Per fer la declaració d'una referència a un objecte de la classe Estudiant haurem de fer:

Estudiant e1;

I per procedir a l'instanciació d'un objecte de la classe Estudiant, utilitzarem **l'operador new** que asignarà memòria a l'objecte i retornarà una referència a aquest objecte (adreça de memòria) i li assignarà a e1:

e1 = new Estudiant(33333333);

Podem crear la **classe testEstudiant** que contindrà el mètode estàtic **main** i presentarà per pantalla diversos exemples de la classe Estudiant:

```
public static void main(String[] args) {
        //Declaració d'una referència a un objecte de la classe Estudiant
        Estudiant e1;
        //Instanciació d'un objecte: new assigna memòria i li dona una referència
        e1 = new Estudiant(33333333);
        //Tot de cop
        Estudiant e2 = new Estudiant();

        System.out.println("DNI: "+ e1.getDni());
        System.out.println(e1);
        System.out.println(e2);
        System.out.println(e1.toString());
        System.out.println(e2.toString());
        e1.setNota(8);
        e2.llegir();
        System.out.println(e1);
        System.out.println(e2);
}
```

Cal destacar que tindrem ara dos fitxers: un per la classe Estudiant, i un altre pel main (testEstudiant o també classMain). Es pràctica habitual en projectes grans fer això (programació modular), malgrat el main sempre es pot posar dins d'una classe.

Noteu que l'accés als membres d'una classe (atributs i mètodes) es realitza mitjançant **l'operador punt (.)** precedit per un objecte de la classe; en l'exemple:

e1.setNota(8), e1 és l'objecte, setNota(8) és el mètode.

El mètode **toString()** és: *@Override* que indica que s'està sobrecarregant (modificant) el seu comportament original i que en aquest cas esta definit a la **classe Object.** Més endavant veurem millor això.

BASE PRÀCTICA

- **Programes d'exemple**

1.- Comproveu la següent aplicació i feu el seu diagrama UML corresponent.

Arxiu CompteBanc.java

```
package comptebanc;

import javax.swing.JOptionPane;

public class CompteBanc {
   private String propietari;
   private Double saldo;

   CompteBanc(String propietari, double saldo){
      this.propietari = propietari;
      this.saldo = saldo;
   }
   public void depositar(double cantidad){
      if(cantidad<0) JOptionPane.showMessageDialog(null, "No es pot retirar saldos negatius");
      else saldo += cantidad;
   }

   public void retirar(double cantidad){
      if(cantidad<0) JOptionPane.showMessageDialog(null, "No es pot retirar saldos negatius");
      else{
         if( (saldo - cantidad) < 0 ){
            JOptionPane.showMessageDialog(null, "No hi ha prou saldo");
         }else saldo -= cantidad;  }
   }
   public double getSaldo(){
      return this.saldo;
   }
}
```

Arxiu Main.java

```
package comptebanc;

import javax.swing.JOptionPane;

public class Main {

   public static void main(String[] args) {
      CompteBanc micuenta = new CompteBanc("Alfons Valverde",1000);

      micuenta.depositar(Double.parseDouble (JOptionPane.showInputDialog(null, "Deposite una
cantidad:")));
      micuenta.retirar(Double.parseDouble (JOptionPane.showInputDialog(null, "Retire una cantidad:")));
      double quantitat = micuenta.getSaldo();
      JOptionPane.showMessageDialog(null, quantitat);
   }
}
```

- **Programes per desenvolupar**

1.- Agafant com a referència el programa fet com a exemple de la classe Estudiant, afegiu un nou atribut: *String nom*, i un nou mètode: double arrodonirNota( ) que arrodoneixi la nota d'un estudiant.

- **Programes opcionals**

1.- Dissenyeu i implementeu una aplicació que a partir de la classe Persona i els seus atributs: nom (String), cognom1 (String), cognom2 (String), dni (int), edad (int), i amb els mètodes que considereu oportuns, demani les dades de dues persones a l'usuari i les mostri per pantalla. Feu el diagrama UML corresponent a la classe dissenyada.